Security improvements in {TeX} Live (Study of Mathematical Software and Its Effective Use for Mathematics Education)

AUTHOR(S):

Preining, Norbert; TEX Live Team

CITATION:

Preining, Norbert ...[et al]. Security improvements in {TeX} Live (Study of Mathematical Software and Its Effective Use for Mathematics Education). 数理解析研究所講究録 2017, 2022: 10-16

ISSUE DATE:

2017-04

URL:

http://hdl.handle.net/2433/231761

数理解析研究所講究録
第2022巻 2017年 10-16

Security improvements in TEX Live

PREINING Norbert
アクセリア株式会社、東京都
TEX Live Team
北陸先端科学技術大学院大学、能美市

Abstract

We report on security improvements in the TEX Live distribution introduced for the release 2016. We present possible attack vectors on older releases, and explain how the recent changes render these attack vectors impossible.

1 Introduction

Since the switch to the current distribution method and the introduction of network installs and updates, many things have changed in the TEX (Live) world. But one thing hasn't kept up with the new distribution methods: security.

There was hardly any verification going on whether the package downloaded from the CTAN mirrors haven't been tampered with. And although we are shipping checksum and file size information, these were only used on rare instances (when restarting a failed installation).

We will recall the general layout of the TEX Live distribution, as well as the security state up till release 2015 and possible attack vectors onto TEX Live installations in Section 1.1. The following Section 2 explains the new security infrastructure and how integrity and authenticity can be guaranteed by it. The following Section 3 is dedicated to implementational and social problems we faced in the process of introducing these changes, as well as general user experience.

1.1 Overview on the TEX Live distribution model

While we cannot give a detailed explanation of the internals, as well as refrain from giving a practical introduction to *using* the TEX Live Manager `tlmgr`, we want to give an overview how packages are installed by `tlmgr` (and similarly by the install script), as this is a necessary prerequisite in understanding the security implications. We assume that the reader has experience in using TEX Live as well as the TEX Live Manager `tlmgr`.

Every TEX Live instance, that is any actual installation on a user's computer, as well as the repositories used to distribute TEX Live, carries all of the relevant information bundled in one file, the so call *TEX Live Database* [4], normally named `texlive.tlpdb`. This file contains general information about the instance, like setting of the used repositories, or distribution format, as well as all contained packages with list of files etc. Thus, this file plays a central role in every operation `tlmgr` is carrying out.

In particular, what happens when a user asks `tlmgr` to update the installation is the following:

1. `tlmgr` loads the local database and determines the set of installed packages,
2. loads the remote database (the one from the used repository)
3. compares the revision numbers and determines packages that need to be update/added/removed,
4. for each package to be updated (or installed) between 1 and 3 so called *containers* (currently `tar.xz` arhcives) are downloaded,
5. each container is unpacked into the destination tree and the local database updated.

1.2 TEX Live security status up to release 2015

Till 2015, the `texlive.tlpdb` at the servers did contain certain consistency information, in particular the size of the containers as well as their checksum (md5), see the following excerpt from a `texlive.tlpdb`:

```
name 12many
...
containersize 2100
containermd5 .....
doccontainersize 375404
doccontainermd5 ....
...
```

This information could be used to at least guarantee that the correct containers have been downloaded, but it turned out that our programs, in particular `tlmgr` and the installation routines did not use these information during normal runs, but only during interrupted installations.

1.3 Possible attack vectors

Improving security brings first a complication of the necessary software, increased workload of the maintainers, changed behavior for users, and thus it is wise to discuss first whether a better security model is necessary. This can be done by scrutinizing the infrastructure for possible attack vectors, and looking at the consequences for the users.

We consider the worst case scenario for the user, namely that a crypto-virus infected the computer and all data is lost.

1.3.1 Attack vector 1

In the simplest case an attack can be carried out by the following steps:

1. compromise one CTAN mirror
2. exchange the `pdftex` binary with one shipping a crypto-virus
3. wait for users to update

It must be noted that worldwide there are a lot of different CTAN mirrors, practically all of them out of our personal control. We cannot even check the security level on these servers, so a possible breach will remain completely unknown to us.

This attack vector could be blocked by actually checking the checksum as provided by the `texlive.tlpdb`. And while changes implementing this feature have been in `tlcritical` (a testing repository for changes to the infra-structure of TEX Live), these changes were never shipped out to users.

1.3.2 Attack vector 2

Another possible attach vector, a bit more involved than the previous, consists of the above steps plus one more:

1. compromise one CTAN mirror
2. exchange the `pdftex` binary with one shipping a crypto-virus
3. adjust the container in a way that the checksum does not change
4. wait for users to update

Here the attacker would byte-pad the container in a way that the checksum is not changed. Note that while this is possible for the checksum algorithm we have been using till now (MD5), the requirement that the resulting file still works as a `tar.xz` archive restricts this attack.

While less likely to happen, this attack vector couldn't be blocked up till release 2016.

1.4 Attack vector 3

Finally, let us consider the most probable attack vector:

1. compromise one CTAN mirror (or set one up yourself!)
2. exchange the `pdftex` binary with one shipping a crypto-virus
3. adjust the checksum in the `texlive.tlpdb`
4. wait for users to update

Indeed, an intruder that can change the containers itself is of course also able to change the data in the `texlive.tlpdb`, thus rendering all protective measures futile. Again, up till release 2016 there was no way to handle this attack vector.

This short list of rather simple attack scenarios gives a clear indication that there is ample way to improve.

2 Security infrastructure

From the previous discussion on possible attack vectors one can extract two requirements of the distribution mechanism necessary to guarantee security: Integrity and authenticity

Integrity of the packages downloaded: Here any kind of tampering should be caught, not only by intruders but also by errors in the download process or disk errors on the server.

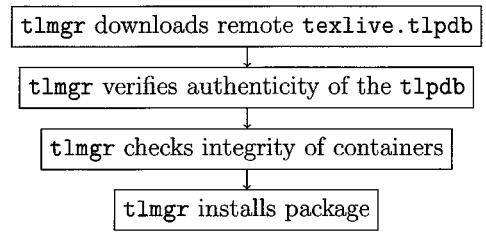
Checksums are normally used for integrity, and because the currently used MD5 is not strong enough, we have switched to SHA512.

Authenticity guarantees that the package downloaded is actually the one that we the TEX Live distributors have prepared, and not someone else.

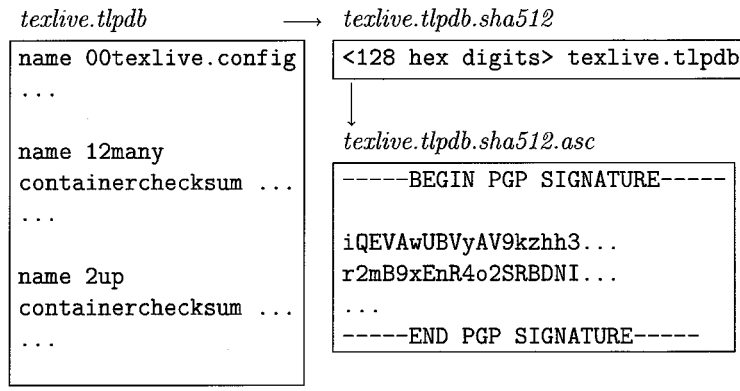
To guarantee authenticity we introduced cryptographic signatures.

2.1 Overview of the verification architecture

The general flow of package download and verification is as follows:



Checking authenticity of the downloaded `texlive.tlpdb` is done by first downloading its respective checksum `texlive.tlpdb.sha512`, as well as the cryptographic signature of the checksum file, named `texlive.tlpdb.sha512.asc`. The GNU Privacy Guard (gnupg) is used to verify that the downloaded signature and the checksum file agree. We refer the reader to introductions to asymmetric cryptography concerning details of this process.



2.2 The signing key

Asymmetric cryptography involves a key-pair with a private and a public key. The private key is used to create signatures, and the public key is used to verify signatures (encryption is another use case, but not relevant to this article). The private key used is (slighly edited output to fit the page):

```
pub   2048R/06BAB6BC 2016-03-19
Key fingerprint=C78B 82D8 C795 12F7 9CC0  D7C8 0D5E 5D91 06BA B6BC
uid       TeX Live Distribution <tex-live@tug.org>
sig 3 06BAB6BC 2016-03-19 TeX Live Distribution <tex-live@tug.org>
sig 3 06BAB6BC 2016-03-19 TeX Live Distribution <tex-live@tug.org>
sig   860CDC13 2016-03-20 Norbert Preining <norbert@preining.info>
sig   30D155AD 2016-03-20 Karl Berry <karl@freefriends.org>
```

It carries, besides the self-signatures, the signatures of Karl Berry and myself, and thus allows easy trust verification since both our keys are available from the usual key servers, as well as in my case the Debian keyring.

As a technical detail, we are not using the actual private key to generate signatures, but a so called sub-key, and keep the main key completely off-line. This guarantees that in case of a breach of the `tug.org` server and compromise of the used signing key, the main key is *not* compromised and can be used to revoke the subkey and generate a new one.

2.3 Discussion

2.3.1 Integrity

Due to the stringent verification of checksums of all downloaded files, the integrity of any packages is guaranteed.

2.3.2 Authenticity of the containers

Authenticity of the downloaded TEX Live Database `texlive.tlpdb` is given by the cryptographic signature. Authenticity of the downloaded containers is guaranteed by the fact that the checksum information for each containers is taken from the (authenticated) database, and the checksum algorithm (SHA512) cannot (at the current time) be tampered with.

3 Practical problems and user experience

As expected, we did face a long list of problems and surprises during the implementation, testing, and transition course. In this last part we discuss a few of them, and how we worked around it, and also report on the user experience.

3.1 (Non-)Distribution of GnuPG

The whole setup is based on the fact that the GNU Privacy Guard GnuPG [2] is available. Without it, authenticity cannot be guaranteed, and as consequence also integrity. Practically all Unix-like distributions ship GnuPG, but there are two notable and important exceptions: Windows and OSX. The default solution – as we do it for other programs commonly available on most platforms but Windows and Mac, e.g., `wget` – would be to ship binaries of GnuPG with TEX Live.

Unfortunately, due to a long history of ignorant legislative bodies, import and export of crytographic software is (might still be) heavily restricted in some countries, governed by the Waasenaar Arrangement [1]. (For those who remember how the first copy of PGP reached Europe, they will understand easily!) And although the situation has changed slightly and export seems to be less restricted nowadays, import into some countries, notably France and India, seems to be a no-go.

As TUG we do *not* want to get involved into legal problems when sending DVDs into foreign countries, so we refrained from distributing GnuPG from within TEX Live.

3.2 Alternative distribution of GnuPG

To mitigate the problems described in the previous section, I personally did set up a repository containing GnuPG binaries for Windows and Mac, which allows users to simply add GnuPG after the initial installation. To obtain GnuPG for either of the mentioned platforms, simply issue the following command

```
tlmgr -repo http://www.preining.info/tlgpg/ install tlgpg
```

On the Mac, the TEX Live Utility (TLU) already offers to install GnuPG from this location on first run, and the TEX Live infrastructure itself is prepared to use the GnuPG installed via this way.

3.3 Computing checksums

Another problem we faced is how to compute checksums, in particular SHA512. There is an excellent and quick Perl module `Digest::SHA` which we initially used, but it turned out the the default Perl distribution even on new Macs is, well, old, very old, and does not ship this module.

We tried a view different options, like pure Perl implementations, or Lua implementations, but all of them were far to slow for the amount of checksums we have to computer.

As a solution we implemented a multi-trial system: First the existence of `Digest::SHA` is checked. If it is not available, the following command line programs are checked in order: `openssl`, `sha512sum`, and `shasum`. One or the other should be available on all platforms.

3.4 Users' complains and experience

As laid out in the introduction, one of the aims was that the user experience should change as little as possible. During the initial testing phase we got quite some complains that our initial implementation and way of warning users was to intrusive, too rough. We have reworked the user experience in a way that, although things are checked and verified as far as possible, only in worst-case scenarios the user is warned, the rest of the changes are minimal.

In particular, for the `tlmgr` calls we now have the following layout:

```
[~] tlmgr update --list --repo <CTAN>/tlnet/
tlmgr: package repository <CTAN>/tlnet/ (verified)
...
```

The only new part is the `(verified)` part, which also might look like

```
(not verified: <reason>
```

where the reasons are either a missing GnuPG, or a missing signature on the server side.

Only in case that a signature was checked and couldn't be verified, the user will receive a stark warning and `tlmgr` aborts.

3.5 Key management

Immediately after release we received request for supporting different signature keys for alternative repositories. I thus have added key management functions, `tlmgr key`, that allows for listing, adding, and removing of additional keys to the trusted list of signing keys.

This feature is already in use by at least the KOMA-Script project [3].

4 Conclusion

With the release 2016 of TEX Live we are now up to the same standard of major Linux distributions which ensure authenticity via cryptographic signatures. Although in the long years of TEX Live being out there, not one case of abuse has been reported, we believe that in the current environment with high risk viruses being distributed in ever evolving schemes, to support verification of authenticity and integrity is a great step forward. A step that most users will not even realize it has been made, but still important.

References

[1] The Wassenaar Arrangement on Export Controls for Conventional Arms and Dual-Use Goods and Technologies. `https://en.wikipedia.org/wiki/Wassenaar_Arrangement`.

[2] Werner Koch et al. GNU Privacy Guard GnuPG. `http://www.gnupg.org/`.

[3] Markus Kohm. KOMA-Script. `http://www.komascript.de/`.

[4] Norbert Preining. TEX Live's new infrastructure. *ArsTEXnica*, 4:69–73, October 2007.